Bond University

Research Repository

Performance comparison of evolutionary algorithms for airfoil design

Randall, Marcus; Rawlins, Tim; Lewis, Andrew; Kipouros, Timoleon

Published in:
International conference on computational science, ICCS 2015 Computational science at the gates of nature

DOI:
10.1016/j.procs.2015.05.384

Licence:
CC BY-NC-ND

Link to output in Bond University research repository.

Recommended citation(APA):
Randall, M., Rawlins, T., Lewis, A., & Kipouros, T. (2015). Performance comparison of evolutionary algorithms for airfoil design. In S. Koziel, L. Leifsson, M. Lees, VV. Krzhizhanovskaya, J. Dongarra, & PMA. Sloot (Eds.), *International conference on computational science, ICCS 2015 Computational science at the gates of nature* (1 ed., Vol. 51, pp. 2267-2276). (Procedia Computer Science). Elsevier. https://doi.org/10.1016/j.procs.2015.05.384

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

For more information, or if you believe that this document breaches copyright, please contact the Bond University research repository coordinator.

Download date: 05 May 2023

Procedia Computer Science

Volume 51, 2015, Pages 2267–2276

ICCS 2015 International Conference On Computational Science

Performance Comparison of Evolutionary Algorithms for Airfoil Design

Marcus Randall[1,2], Tim Rawlins[2], Andrew Lewis[2], and Timoleon Kipouros[3]

[1] Department of Informatics
Bond University, QLD 4229, Australia
Ph: +61 7 55953361
Email: mrandall@bond.edu.au
[2] Institute for Integrated and Intelligent Systems
Griffith University, Brisbane, Australia
[3] Department of Engineering
University of Cambridge, Cambridge, United Kingdom

Abstract
Different evolutionary algorithms, by their very nature, will have different search trajectory characteristics. Understanding these particularly for real world problems gives researchers and practitioners valuable insights into potential problem domains for the various algorithms, as well as an understanding for potential hybridisation. In this study, we examine three evolutionary techniques, namely, multi-objective particle swarm optimisation, extremal optimisation and tabu search. A problem that is to design optimal cross sectional areas of airfoils that maximise lift and minimise drag, is used. The comparison analyses actual parameter values, rather than just objective function values and computational costs. It reveals that the three algorithms had distinctive search patterns, and favoured different regions during exploration of the design space.

Keywords:

Keywords: comparative analysis, tabu search, particle swarm optimisation, extremal optimisation, multi-objective optimisation, airfoil design.

1 Introduction

In many papers it is often the case that a single evolutionary optimisation algorithm is developed or refined and then applied to a problem or a suite of test problems. While many of these do indeed make some comparison with standard algorithms (Non-Dominated Sorting Genetic Algorithm (NSGA) [7] being a famous example for multi-objective problems), this is usually in the form of the overall best objective costs achieved by the algorithms, as well as, more occasionally, computational runtime. This is certainly a valid way of assessing overall effectiveness of an algorithm and has lead to many valuable insights.

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

© The Authors. Published by Elsevier B.V.

doi:10.1016/j.procs.2015.05.384

This paper attempts to delve deeper into the relative performances of evolutionary algorithms by examining the characteristics of the search space parameter values that evolutionary algorithms are capable of obtaining. A graphical device known as a "parallel coordinate plot" [12] is used to visually aid understanding of parameter value exploration across algorithms. Our implementation of this comparative technique involves three well known algorithms, particle swarm optimisation, tabu search, and extremal optimisation on the real world problem of designing 2D airfoil sections that aim to minimise drag, but maximise lift potential.

The remainder of the paper is organised as follows. Section 2 describes each of the algorithms used in the comparison while Section 3 given an introduction to the real-world problem. Section 4 details how each algorithm as run and an analysis of their relative performances against one another. Finally, Section 5 explains how these results may be leveraged in future research.

2 Evolutionary Algorithms for Multi-Objective Problems

In this study, we use three evolutionary optimisation algorithms that all operate in very different ways. In this section, a brief description of each and its application to multi-objective optimisation problems, is given.

2.1 Multi-Objective Particle Swarm Optimisation (MOPSO)

MOPSO is a multi-objective extension of Particle Swarm Optimisation (PSO) that itself models a simplified version of bird flocking [14]. The essential idea being that the collective effort of the birds (who influence one another) will discover food sources better than a single bird might. In the algorithmic versions, the addition of an inertia weight term of Shi and Eberhart [20] balances the role of local and global search. The typical implementation of a standard PSO can be visualised as a swarm of particles that move through (n-dimensional) parameter space guided by a) the location of the global best result found by the swarm, b) their individual memories of the location of their personal best results and c) by their inertia.

MOPSO is an extension of PSO in which objective space is also n-dimensional, i.e., there are two or more (often competing) objectives to optimise. A consequence of this is that there is no longer a single "best" solution but rather a set of trade-off solutions that are referred to as Pareto-optimal solutions [6]. Extending PSO to multiple objectives is achieved by adding an archive of Pareto-dominant solutions from which a global "best" solution was chosen using roulette wheel selection. In addition, the personal best was modified to be a single non-dominated solution [4].

Each particle in the swarm has a position and a velocity. The equations of motion that govern the movement of a particle in a swarm are:

$$\begin{aligned} V_{t+1} =& (\omega \times V_t) + R_1 \times c_1 \times (PBest_t - X_t) \\ &+ R_2 \times c_2 \times (GBest - X_t) \end{aligned} \tag{1}$$

$$X_{t+1} = X_t + V_{t+1} \tag{2}$$

where V is the particle velocity, ω is inertial weight, R_1 and R_2 are random values between 0 and 1, c_1 is the cognitive component weight (the weight is given to the particles' own search

results), c_2 is the collective component weight (the weight given to the swarm's search results), $PBest$ is the "best" result found by the particle, $GBest$ is a randomly selected member of the archive of best results found by the swarm and X is the particle position.

2.2 Multi-Objective Tabu Search (MOTS)

MOTS (Multi-Objective Tabu Search) [13] builds on a traditional optimisation algorithm, Tabu Search (TS) [9]. Based on the general tenets of it and the work of Connor and Tiley [5], which in turn uses a Hooke and Jeeves (H&J) [11] move as the neighbourhood operator, MOTS implements three levels of tabu memory: short term, medium term and long term. For the former, the search is strictly prohibited from visiting the points in this list. The near-optimal points stored in the medium term memory are used for periodical intensification while information about the regions that have already been explored is stored in the long term memory. These three memories balance the needs of local search and a sensible coverage of the search space as a whole.

In terms of the multi-objective aspect of the algorithm, MOTS adds to the previous work in the following three ways:

- Search point comparison – Pareto dominance techniques are used to compare neighbourhood points.
- The H&J move – As might be expected, the neighbourhood of H&J moves are each made, and the ones that are not tabu or violate the constraints become the candidate set. These are sorted in terms of Pareto dominance. The possibility of multiple points being Pareto-equivalent and optimal is allowed for by classifying each candidate point according to its domination or Pareto-equivalence to the current point. If there is a single dominating point, it is automatically accepted as the next point. If there are multiple dominating points, the dominated points within that group are removed and one is selected at random from those remaining. In essence, the next point is simply the"best" allowed point, be that uphill or downhill, from the candidate solutions.
- Optimal point archiving and the medium term memory – An unbounded list of non-dominated points form the archive. These are the best trade-off solutions that have been generated and correspond to the medium term memory.

2.3 Multi-Objective Extremal Optimisation (MOEO)

Canonically, Extremal optimisation [1, 2, 3] manipulates a single solution for single objective optimisation problems. The main characteristics of the technique are that it does not converge and, at each iteration, a probabilistic worst solution component's value is changed to a random value. Solution components are the building blocks of the solution, an example being a city placed between a preceding and subsequent city in a chain that forms a TSP tour. In the original version of the EO algorithm, at each iteration the component whose fitness was worst would be replaced by another solution component value generated at random. In essence, however, this choice of always selecting the worst component to modify led to a search that was too greedy, and consequently its performance was poor. Like other meta-heuristic algorithms, an element of randomness (in the form of the probabilistic selection of solution components to change) was introduced. This became known as $\tau-$EO [3]. Components are ranked from worst (rank 1) to best (rank n). The parameter τ and the rank control the selection probability for each solution component [2]. This is achieved using Equation 3.

$$P_i \propto i^{-\tau} \qquad 1 \leq i \leq n \tag{3}$$

Where:

i is the rank of the component,

P_i is the probability ($P_i \in [0, 1]$ when normalised) that component i is chosen and

n is the number of components.

The question becomes: how are components in a multi-objective optimisation problem ranked? To complicate this, the problem contained herein is a blackbox function evaluation, which means there is no way of calculating any delta change values. As explained in Randall et al. [19] each solution component is mutated and then its subsequent function evaluations are ranked using dominance ranking. Effectively, there are n new solutions, where n is the size of the solution vector. The first ranked solution is most dominated by the others, and according to EO rules, has the highest probability of being chosen.

3 Optimisation of Airfoil at Stall Angle

This is a real-world, continuous free form deformation (FFD) problem that has a vector of eight variables in which the lift objective is to be maximised[1] while drag is to be minimised [16]. This is hereafter referred to as the 8-parameter FFD (FFD-8) problem. This problem has an 8-dimensional parameter space, corresponding to the change in x and y positions of the 4 central control points of the 8 point free form deformation technique used to manipulate the airfoil shape. Figure 3 shows the 8 control points, with the 4 central points marked, and with their possible movements shown. The 8 parameters can be viewed as 4 pairs in order, corresponding to the labelled control points in order, the first value in each pair represents horizontal movement, while the second is vertical, positive values correspond to right and upward movement from the initial position respectively, while negative values correspond to left and downward movement. The objectives were to maximised lift and minimise drag, as calculated by Xfoil [8]. To simplify the calculation, negative lift is instead minimised. Many solutions to this problem are considered infeasible due to violating design constraints or not functioning at an angle of attack of 15 degrees ("stall angle".). Moreover, this is a blackbox optimisation problem in which each function evaluation may take up to a minute of computational time on standard Pentium processor.

This problem is reasonably complex to calculate and can serve as a benchmark for more complex aerodynamics problems.

4 Comparing Performance Across Algorithms

This section describes how the values of the individual algorithm control parameters are configured. Beyond this, the comparative analysis is given, highlighting the the characteristics of the tendencies and strengths of each. In order to compare the algorithms fairly, each is allowed 6000 function evaluations. While this may seem relatively few, as explained in the previous section each evaluation is relatively computationally expensive. Performance is tracked across all solutions, feasible or infeasible, regardless of whether they enter the multi-objective archive or not.

[1] This objective is often minimised by multiplying its value by -1 [10].

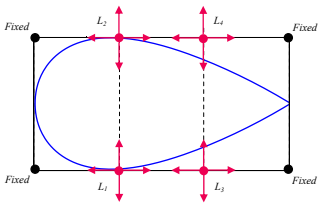

Figure 1: Schematic of 2D airfoil and FFD structure

4.1 Experimental Design

For our MOPSO implementation, we implemented a MOPSO based on the original proposal [4]. Control parameters are given in Table 1. We use 60 particles and 100 iterations. 60 particles were used specifically because past experimentation has suggested that this is a value with good general performance.

Table 1: Summary of parameters values used.

Parameter	**Value**
Particles	60
Iterations	100
Intertial Weight	0.4
Cognitive Weight Coefficient	2
Collective Motion Weight Coefficient	2
Archive Size	60
Problem Specific	
Initial Parameter Generation	Uniform from -1.0 to -1.0
Parameter Constraints	Constantly constrained to between -1.0 to 1.0

The parameter settings for MOEO are the same as used for Randall et al [19] and the reader is referred to this paper for a full explanation of them. Specifically, there are 100 test local search points per iteration and a diversification probability of 0.5 if local search fails. In terms of MOTS, the parameter values are from Kipouros et al [15] and are: $nstm = 15$, $nregions = 4$, $intensify = 15$, $diversify = 25$, $restart = 45$, $stepsize = 0.07$, $stepsizerf = 0.5$ and $nsample = 8$.

4.2 Comparative Results

A number of parallel co-ordinate plots were generated from the output of the the three algorithms. Figure 2 gives the plot of the feasible results of all the algorithms. It should be noted that these plots are from a single run of each algorithm, with each given a budget of 6000 function evaluations. Single runs were compared to try to appreciate the individual behaviour of the algorithms, rather than getting a statistical combination of multiple runs that might "blur" the details. It is possible that another run for an algorithm may have given "better" results, but this could be said for any of the algorithms. What is of interest is how well each algorithm

performs within a run – how effective is its algorithmic advantage compared to random search. To the extent possible, the runs selected were "representative". This is complicated by the fact that at this stage the comparisons have been qualitative. It may be possible to try to generalise the observations but in the absence of quantitative metrics the degree of similarity across different runs is difficult to judge. This remains a topic for further investigation.

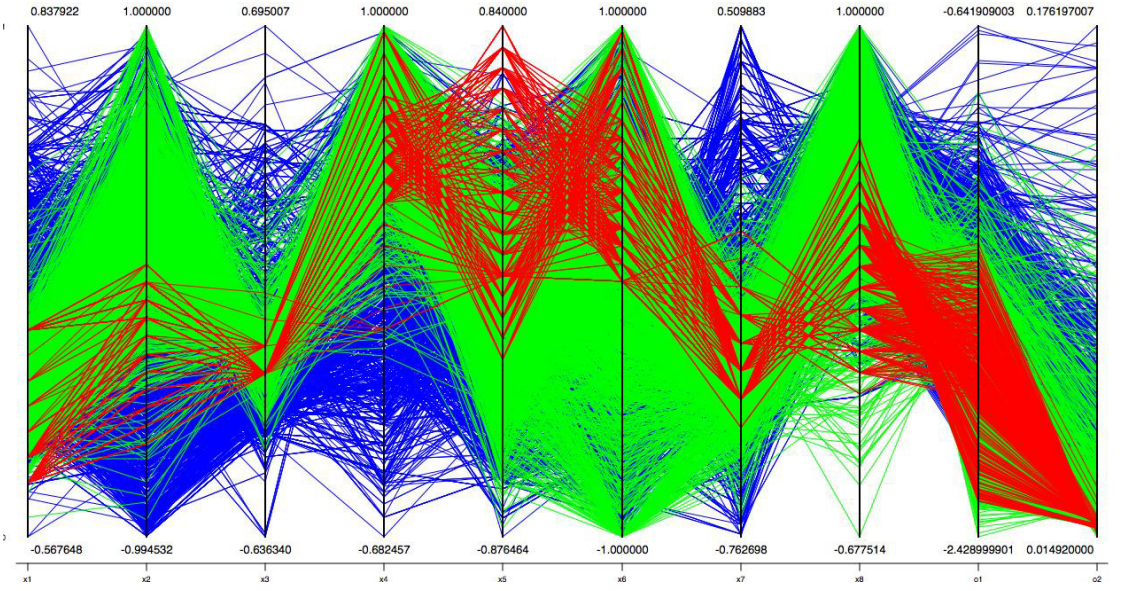

Figure 2: Parallel Coordinates plot of all feasible solutions for all algorithms tested, MOTS (red), MOPSO (green) and MOEO (blue)

From inspection of Figure 2 we can infer the following. MOTS is very structured in its exploration of the design space. It appears almost to be combining discrete parameter values in changing combinations. It zeros in on some quite good values, but MOPSO is achieving comparable results - MOTS gives better values for objective 2, MOPSO better for objective 1. It is quite apparent that MOPSO has explored a greater range of parameters than MOTS.

MOEO behaves in a similar manner to MOPSO – both are not inhibited by infeasible space, having mechanisms to traverse it into feasible regions. On several parameters it has explored a greater range of values than MOPSO. This does not, however, translate into finding better values. As can be seen from the plot of feasible values for MOEO and MOPSO, Figure 3, MOPSO outperforms MOEO on each objective, and on "compromise" solutions. (The latter point can be seen by careful inspection – there are blue (MOPSO) lines horizontal below any comparable MOEO "compromise" solution: all MOEO solution pairs for the two objectives are at higher values, or greater slope, than the MOPSO solutions.)

The reasons for these differences might be due to the greater propensity for MOPSO to exploit known good solutions to converge to minima, due to the algorithm following good solutions – acting at the front of the swarm – instead of removing poor solutions – acting at the back. This suggests that the benefits of both algorithms might be obtained by combining their mechanisms in a hybrid algorithm. The hybridisation could employ an approach similar to that

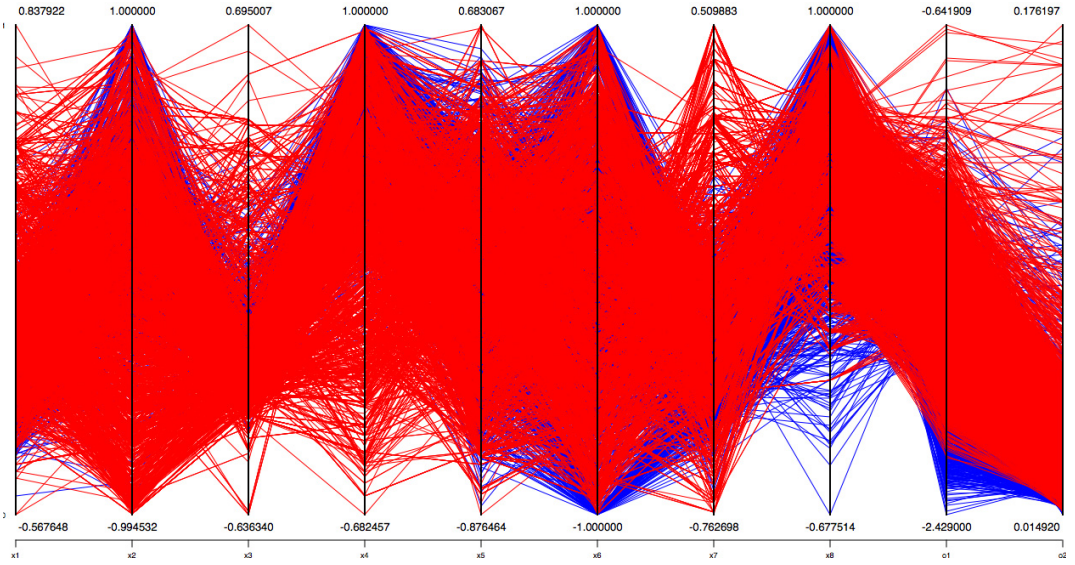

Figure 3: Parallel Coordinates plot of feasible solutions for MOEO (red) and MOPSO (blue)

proposed by Lewis et al. [17], using Evolutionary Population Dynamics (EPD) to improve the population of the MOPSO algorithm using operations that mirror those of MOEO.

The two plots of MOEO and MOPSO for feasible solutions and for all data, including infeasible, show the exploration capability of these two algorithms. Figure 4 shows an interesting quirk – MOEO appears to have missed exploring almost half the range of parameter 8. Examination of the feasible solutions obtained, in Figure 5, shows MOPSO, for this run, failed to comprehensively explore parts of the range of values of parameters 3, 4 and 7. It did not fail to explore parts of the ranges completely – there are still some random samplings of them – but the coverage is of reduced density. It also appears that the parameter values in the range that MOEO missed for parameter 8 gave rise to very few feasible solutions – there are far fewer feasible solutions MOPSO found (Figure 5) than the total, including infeasible solutions (Figure 4).

All of the algorithms performed fairly well and reasonably comparably. MOPSO and MOTS were very much comparable, in terms of the results produced. MOEO was a (very) little behind them. But, MOTS achieved its results with a less than comprehensive coverage of the design space – if there had been really good results lurking out in the unexplored regions, could we have been assured of finding them? MOPSO in part answers the question, in that it appears to have gone and looked – its mechanism gives more assurance of not missing good results, especially in complex design spaces with significant regions of infeasibility. This may be dependent on a good initial distribution (sampling).

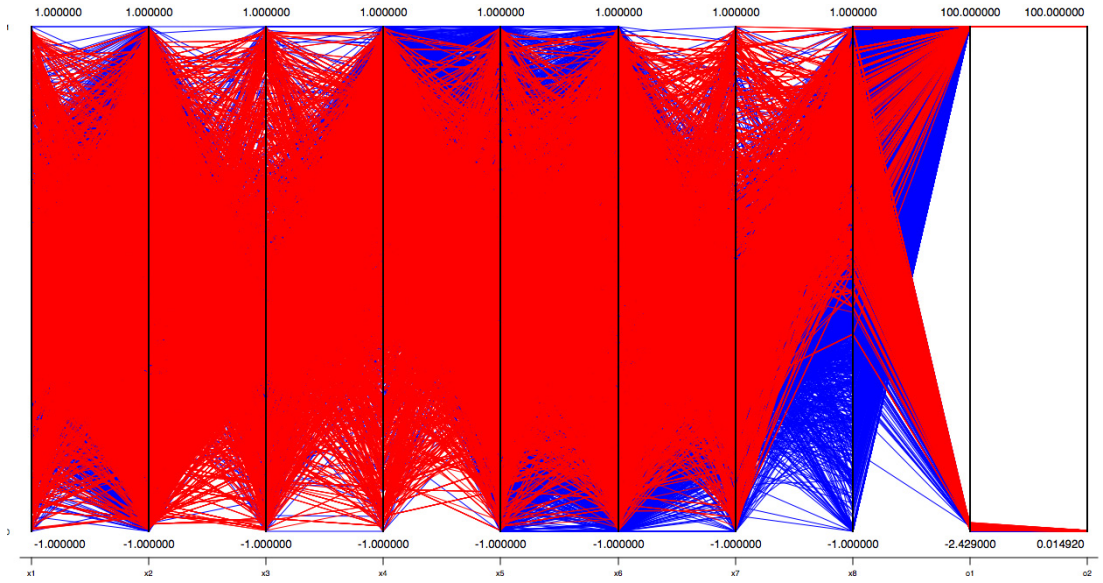

Figure 4: Parallel Coordinates plot of all solutions for MOEO (red) and MOPSO (blue)

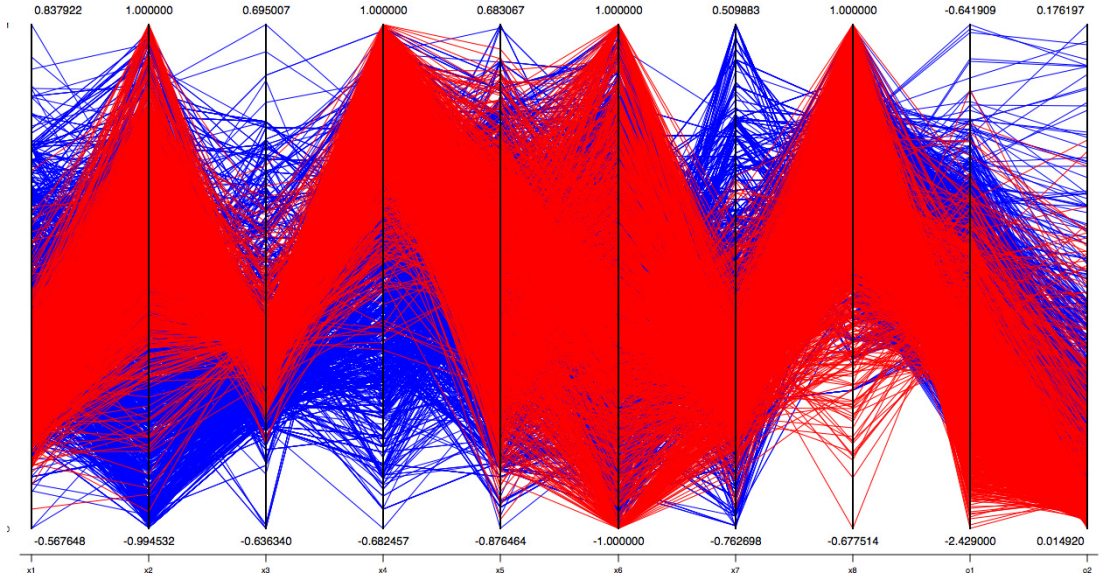

Figure 5: Parallel Coordinates plot of feasible solutions for MOEO (red) and MOPSO (blue)

5 Conclusions

Understanding the behaviour of evolutionary algorithms, particularly the states they explore and trajectories they take, allows us to better design appropriate implementations. In this first of its kind study, it was very evident that the three algorithms favoured different extents of explorations on the different parameters. Thus, the concept of hybridisation of MOPSO and MOEO by using Evolutionary Population Dynamics [18] appears promising and will be explored in future comparative investigations of these and further algorithms.

However, before this more informed hybridisation can begin, we need to do further experimentation in terms of different problem types, and the use and analysis of more problem instances. In particular, the differing abilities of algorithms in handling infeasible regions of search spaces need to be tested, while keeping the studies firmly grounded in the solution of real-world problems. This is not to mention the use of other evolutionary techniques such as differential evolution. It is anticipated that this will require substantial amounts of computational time, but should yield solvers that are able to more thoroughly exploit and explore complex, real-world, search spaces.

References

[1] S. Boettcher and A. Percus. Extremal optimization: Methods derived from co-evolution. In *Proceedings of the Genetic and Evolutionary and Computation Conference*, pages 825–832. Moran Kaufmann, 1999.

[2] S. Boettcher and A. Percus. Nature's way of optimizing. *Artificial Intelligence*, 119:275 – 286, 2000.

[3] S. Boettcher and A. Percus. Extremal optimization: An evolutionary local search algorithm. In H. Bhargava and N. Ye, editors, *Computational Modeling and Problem Solving in the Networked World*, Interfaces in Computer Science and Operations Research, pages 61–77. Kluwer Academic Publishers, 2003.

[4] C. Coello Coello and M. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1051 –1056, 2002.

[5] A. Connor and D. Tilley. A tabu search method for the optimisation of fluid power circuits. *Journal of Systems and Control*, 212:373–381, 1998.

[6] K. Deb. Multi-objective optimization. In E. Burke and G. Kendall, editors, *Search Methodologies*, pages 273–316. Springer US, 2005.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[8] M. Drela and H. Youngren. Xfoil, subsonic airfoil development system, 2008. http://web.mit.edu/drela/Public/web/xfoil/.

[9] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[10] J. Hetenhausen, A. Lewis, M. Randall, and T. Kipouros. Interactive multi-objective particle swarm optimisation using decision space interaction. In *Proceedings of the Congress of Evolutionary Computation*, pages 3411–3418, 2013.

[11] R. Hooke and T. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM*, 8:212–229, 1961.

[12] A. Inselberg. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Springer, New York, 2009.

[13] D. Jaeggi, G. T. Parks, T. Kipouros, and P. J. Clarkson. The development of a multi-objective tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research*, 185(3):1192–1212, 2008.

[14] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942 –1948 vol.4, nov/dec 1995.

[15] T. Kipouros, T. Peachy, D. Abramson, and M. Savill. Enhancing and developing the practical optimisation capabilities and intelligence of automatic design software. 2012.

[16] V. Lattarulo, T. Kipouros, and G. Parks. Application of the multi-objective alliance algorithm to a benchmark aerodynamic optimization problems. In *Proceedings of the Congress of Evolutionary Computation*, pages 3182–3189, 2013.

[17] A. Lewis, S. Mostaghim, and M. Randall. Evolutionary population dynamics and multi-objective optimisation problems. In L. Bui and S. Alam, editors, *Multi-Objective Optimization In Computational Intelligence: Theory And Practice*, pages 185–206. IGI Global, Hershey, PA, 2008. (1 citation).

[18] A. Lewis, S. Mostaghim, and M. Randall. Evolutionary population dynamics and multi-objective optimisation problems. In *Multi-Objective Optimization in Computational Intelligence: Theory and Practice*, pages 189–210. IGI Global, 2008.

[19] M. Randall, A. Lewis, J. Hettenhausen, and T. Kipouros. Local search enabled extremal optimisation for continuous inseparable multi-objective benchmark and real-world problems. *Procedia Computer Science*, 29(0):1904 – 1914, 2014.

[20] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69 –73, may 1998.